**TASK 1:**

**Video Streaming Application**

**Overview**

This Django-based video streaming application allows users to manage video content, perform CRUD operations on videos, and access video streaming functionalities. The application provides a RESTful API for interacting with video resources.

**Usage**

**Authentication**

To access protected API endpoints, you need to authenticate using token-based authentication.

1. **Obtain Token**
   * Create a user and obtain a token for authentication.
2. **Include Token in Requests**
   * Set the **Authorization** header in your HTTP requests:
   * Authorization: Token <your\_token>

**API Endpoints**

The application exposes the following API endpoints:

* **GET /api/videos/**
  + Retrieve a list of all videos.
* **POST /api/videos/**
  + Create a new video entry.
  + Request Body:

{

"name": "Video Name",

"video\_url": "https://example.com/video.mp4"

}}

* **GET /api/videos/<id>/**
  + Retrieve details of a specific video by ID.
* **PUT /api/videos/<id>/**
  + Update details of a specific video by ID.
  + Request Body:

{

"name": "Updated Video Name"

}

* **DELETE /api/videos/<id>/**
  + Delete a specific video by ID.

**Testing**

Run the test suite to ensure application functionality:

{

"name": "Updated Video Name"

}

**TASK 2:**

# Production Log Management API (OEE)

This Django application manages production logs for machines and calculates Overall Equipment Effectiveness (OEE) based on production metrics.

**Overview:**

1. **Purpose**: This Django application manages production logs for machines and calculates Overall Equipment Effectiveness (OEE) based on production metrics.
2. **Components**:
   * **Models**: **Machine** and **ProductionLog** models represent manufacturing machines and their production logs, respectively.
   * **Serializers**: **MachineSerializer** and **ProductionLogSerializer** define how model data should be serialized/deserialized for API interactions.
   * **Views**: **ProductionLogViewSet** contains API endpoints for CRUD operations on **ProductionLog** objects.
   * **Tests**: **ProductionLogTestCase** includes unit tests for **ProductionLog** model methods and API endpoint functionalities.

**Code Breakdown:**

1. **Models** (**models.py**):
   * **Machine** model stores machine details like name, serial number, and creation time.
   * **ProductionLog** model records production details such as cycle number, material, machine association, timestamps, durations, and production metrics.
2. **Serializers** (**serializers.py**):
   * **MachineSerializer** and **ProductionLogSerializer** define how **Machine** and **ProductionLog** models should be serialized for API responses and deserialized for request data.
3. **Views** (**views.py**):
   * **ProductionLogViewSet** is a Django REST Framework viewset providing CRUD operations (**list**, **create**, **retrieve**, **update**, **destroy**) for **ProductionLog** objects.
   * Custom methods **perform\_create** and **perform\_update** ensure OEE calculation (**calculate\_oee()**) is triggered upon creation/update of **ProductionLog** instances.
4. **Tests** (**tests.py**):
   * **ProductionLogTestCase** includes unit tests:
     + **test\_oee\_calculation**: Tests the **calculate\_oee()** method of **ProductionLog** model.
     + **test\_filter\_by\_machine**: Tests filtering production logs by **machine\_id**.
     + **test\_filter\_by\_date\_range**: Tests filtering production logs by date range.
     + **test\_combined\_filters**: Tests combined filtering by **machine\_id** and date range.

**Workflow:**

1. **Creating Production Logs**:
   * Use Django REST API endpoints (**/production\_logs/**) to create, retrieve, update, or delete production logs (**ProductionLog** instances).
2. **OEE Calculation**:
   * Upon creating or updating a **ProductionLog**, the **calculate\_oee()** method is automatically triggered to compute the OEE based on availability, performance, and quality metrics.
3. **API Endpoints**:
   * **/production\_logs/**: Supports various query parameters (**machine\_id**, **start\_date**, **end\_date**) for filtering production logs.

**API Endpoints**

* **GET /production\_logs/**: List all production logs or filter logs by query parameters:
  + **machine\_id**: Filter logs by machine ID.
  + **start\_date**: Filter logs starting from a specific date (YYYY-MM-DD).
  + **end\_date**: Filter logs up to a specific date (YYYY-MM-DD).
* **POST /production\_logs/**: Create a new production log.
* **GET /production\_logs/<log\_id>/**: Retrieve details of a specific production log.
* **PUT /production\_logs/<log\_id>/**: Update details of a specific production log.
* **DELETE /production\_logs/<log\_id>/**: Delete a specific production log.

## Testing

Run unit tests to ensure functionality:

python manage.py test